AFRL-VA-WP-TP-2006-340

MULTIPLE UAV TASK ALLOCATION FOR AN ELECTRONIC WARFARE MISSION COMPARING GENETIC ALGORITHMS AND SIMULATED ANNEALING (PREPRINT)

Marjorie A. Darrah, William Niland, and Brian Stolarik

AUGUST 2006

Approved for public release; distribution is unlimited.

STINFO COPY

AIR VEHICLES DIRECTORATE
AIR FORCE MATERIEL COMMAND
AIR FORCE RESEARCH LABORATORY
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7542

NOTICE AND SIGNATURE PAGE

AFRL-VA-WP-TP-2006-340 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

*//Signature//

Corey J. Schumacher
Senior Aerospace Engineer
Control Design and Analysis Branch
Air Force Research Laboratory
Air Vehicles Directorate

//Signature//

Brian Gamble
Deputy Chief
Control Design and Analysis Branch
Air Force Research Laboratory
Air Vehicles Directorate

//Signature//

JEFFREY C. TROMP
Senior Technical Advisor
Control Sciences Division
Air Vehicles Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

*Disseminated copies will show "//Signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
August 2006	Conference Paper Preprint	05/01/2006 – 08/01/2006

4. TITLE AND SUBTITLE
MULTIPLE UAV TASK ALLOCATION FOR AN ELECTRONIC WARFARE MISSION COMPARING GENETIC ALGORITHMS AND SIMULATED ANNEALING (PREPRINT)

5a. CONTRACT NUMBER
FA8650-04-C-3402

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602201

6. AUTHOR(S)
Marjorie A. Darrah, William Niland, and Brian Stolarik

5d. PROJECT NUMBER
A052

5e. TASK NUMBER

5f. WORK UNIT NUMBER
0B

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Scientific Research, Inc.
Fairmont, WV 26554

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Vehicles Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson Air Force Base, OH 45433-7542

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL-VA-WP

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-VA-WP-TP-2006-340

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
If this work is published, the publisher may assert copyright. This work was funded by Department of the Air Force Contract FA8650-04-C-3402. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government.

This paper was submitted to the Proceedings of the 26th American Control Conference (ACC 2007), published by IEEE.

PAO Case Number: AFRL/WS 06-2067 (cleared August 25, 2006). Paper contains color.

14. ABSTRACT

This paper compares two algorithms applied to the task allocation of multiple Unmanned Aerial Vehicles (UAVs) for an electronic warfare mission. The electronic warfare mission scenario is discussed and a review of both the genetic algorithm and simulated annealing algorithm is given. The encoding of the problem and the functions and operations needed to implement each algorithm is outlined and compared. The algorithms were implemented and tested in Matlab. A discussion of the performance analysis for the time to convergence and quality of solutions in a fixed period of time is given.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Corey Schumacher
Unclassified	Unclassified	Unclassified	SAR	14	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Multiple UAV Task Allocation for an Electronic Warfare Mission Comparing Genetic Algorithms and Simulated Annealing

Marjorie A. Darrah, William Niland and Brian Stolarik

***Abstract*—This paper compares two algorithms applied to the task allocation of multiple Unmanned Aerial Vehicles (UAVs) for an electronic warfare mission. The electronic warfare mission scenario is discussed and a review of both the genetic algorithm and simulated annealing algorithm is given. The encoding of the problem and the functions and operations needed to implement each algorithm is outlined and compared. The algorithms were implemented and tested in Matlab. A discussion of the performance analysis for the time to convergence and quality of solutions in a fixed period of time is given.**

I. Introduction

The Electronic Warfare (EW) scenario investigated in our research assumes that a number of air vehicles are used to attack enemy air defense targets. The location of potential targets is detected and vehicles are sent to classify the target as a threat or non-threat. If the target is classified as a threat, then a vehicle must attack the target. We assume that there is a circular threat zone around each enemy air defense target and that if a UAV breeches this zone it can be fired on. In order to get into range for the attack, we assume that a jamming vehicle must be engaged to reduce the capability of the threat to track the attacking vehicle. When the threat tracking capabilities have been reduced, the attacking vehicle can proceed with the attack task. After the attack, a vehicle must verify that the threat has been destroyed.

This mission described in the paper has the following basic assumptions:

1. A priori knowledge of the battle space. Possible target locations are known before mission begins.
2. There are four tasks to perform on each target: classify, jam, attack, and verify.
3. The vehicles are all Unmanned Combat Aerial Vehicles (UCAVs) but are outfitted with different payloads.
4. Vehicles have sensors that can allow them to accomplish the classify task out of harms way before jamming starts.
5. Vehicles outfitted with jamming equipment are only tasked to do jamming.
6. Vehicles with weapons payloads also have sensors for classifying and doing battle damage assessment on targets.

This paper explores how a genetic algorithm and simulated annealing algorithm can be used to find a solution to the task assignment of the UAVs in the EW mission. The algorithm approaches are summarized and then similarities are explored. The algorithms are implemented in MATLAB and compared.

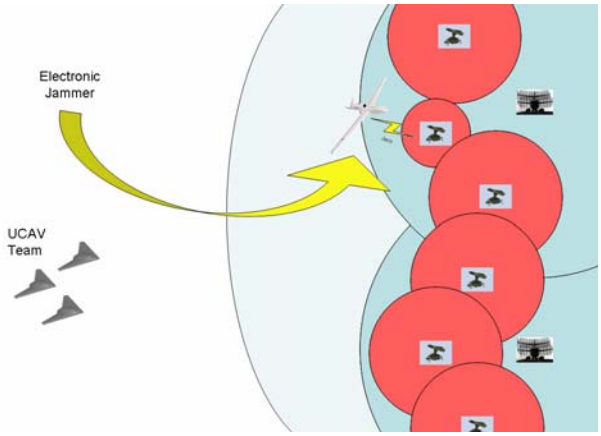

Fig. 1. This is a depiction of an electronic warfare mission. Multiple vehicles are sent to attack Surface-to-Air Missile (SAM) sites. The circles around the sites represent electronic tracking threat zones. The electronic jammer is sent in to reduce the threat zone so that vehicles can perform tasks without being tracked and fired on.

II. Methodology

A. Review of Genetic Algorithms

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest with structured randomized information exchange to form a search algorithm that efficiently exploits historical information to speculate on new search points with expected improved performance [1].

A genetic algorithm can be used to optimize a function by searching a population of points using probabilistic transitions rules. There are several basic functions and operations that make up a genetic algorithm: the objective function, the fitness function, population generation, elitism, reproduction, and mutation.

Genetic algorithms require the parameter set be encoded as a finite-length string over a finite alphabet; these strings are referred to as chromosomes. The entire set of all

Manuscript received September 15, 2006. This research was sponsored by AFRL under contract number FA8650-04-C-3402.

Marjorie A. Darrah, William Niland and Brian Stolarik are with the Institute for Scientific Research, Inc., Fairmont, WV 26554 USA; e-mail: mdarrah@isr..us, wniland@isr.us, bstolarik@isr.us.

possibilities give the entire population for the problem. The genetic algorithm begins by randomly generating a subset of this entire population to be the initial generation. The chromosomes are generated in a way to ensure that each is a feasible solution.

After the initial generation is created then reproduction, elitism, and mutation are applied to produce subsequent generations that, when implemented correctly, have improved performance. The reproductive operators may be implemented in many different ways but involve simple copying and swapping of strings between two parent members to produce a child member. The elitism operator retains members of a generation with top performance so that these members may remain in the next generation and continue to reproduce. The mutation operator is applied to a small percentage of individual members of a generation in order to ensure that the population continues to evolve. The process of producing new generations continues for a set duration or, if time allows, until convergence to the best solution is attained.

B. Review of Simulated Annealing

Simulated Annealing is a method that is based on the analogy of thermodynamics; specifically the way metal cools and anneals [2]. At high temperatures, the molecules of a metal move freely, but as the metal is cooled the thermal mobility is lost. The atoms are often able to line themselves up and reach a minimum energy state for this system. If a liquid metal is cooled quickly it does not reach this state but rather ends up in a polycrystalline or amorphous state with somewhat higher energy. The essence of the slow cooling process is to ensure that a low energy state is achieved by allowing ample time for redistribution of atoms as they lose mobility.

Theoretically speaking, simulated annealing is an iterative improvement algorithm that can be used to effectively solve large scale optimization problems [3]. The goal of simulated annealing is to reach a globally optimal solution without getting stuck in the local minima. The algorithm iterates over a temperature variable, which decreases at a rate defined by the cooling schedule. There are several basic operations and functions that make up the simulated annealing algorithm: the objective function, the cooling schedule, and the neighborhood defining operation.

To begin the simulated annealing process a solution to the problem is constructed. During each iteration, a potential new solution is created by making small changes to the current best solution. The current and new solutions are evaluated. If the new solution is better than the current best solution, then the new solution replaces the current solution. If the new solution is worse than the current best solution, then the new solution may still replace the current best solution with some probability. The probability is based on the temperature function that determines the likelihood of the worse solution being used to replace the current best solution. At a higher temperature, the algorithm will more likely accept the worse solution than it will at a lower temperature.

The simulated annealing algorithm is guaranteed to converge in a finite time if the cooling schedule is sufficiently slow [3]. However, the finite cooling schedule may take too long to be practical in implementation, so applications may sacrifice convergence for a faster cooling schedule. This leads to suboptimal solutions, which may be acceptable in some problem domains.

Both algorithms can be used to address the problem of task assignment for a team of UAVs to conduct an EW mission. The algorithms can utilize the same solution encoding scheme and objective function. The same operations can be used for both to create new solutions from old ones.

C. UAV Task Assignment Problem

In this EW scenario, the UAVs are required to perform four tasks on each of the targets. Let T = {1, 2, … Nt} be the set of targets found and V = {1, 2, …Nv}be the set of UAVs performing the tasks. The set of tasks to be performed is M = {classify, jam, attack, verify} with Nm the number of tasks.

The classification task is performed when a vehicle follows a trajectory that allows it to place its sensor footprint over the target. The classification task can be done from an area outside of the threat zone of the target. The attack task requires that a UAV position itself inside the target threat zone to release a weapon at the target. Before the attack can be accomplished the jamming vehicle must obtain the attacking vehicle's trajectory and begin jamming the radar site emission accordingly. This jamming continues until the attacking vehicle can position itself close enough to launch a weapon at the site and then fly out of harms way to avoid detection or counter attack. Following the attack, damage verification is performed again by placing the UAV's sensor footprint over the target site. Again the verification task can be done outside the threat zone of the target. The jamming vehicle is only utilized when attacking a target. The jamming engagement occurs β seconds before the attack and lasts for γ seconds after the attack to ensure the safety of the attacking vehicle. Fig. 2 depicts how the jamming task and attack task overlap.

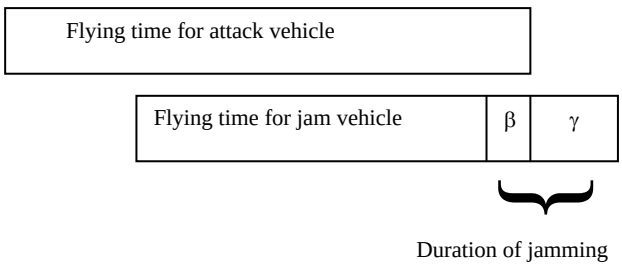

Fig. 2. Attack and Jamming Task Overlap

The assignment algorithm must take into account task

precedence and task coordination. Task precedence requires that all tasks performed on a target must be performed in order (i.e. classify, jam, attack, verify). Task coordination requires that each task be performed once on each target.

D. Encoding the Problem

In order to begin the problem, possible solutions need to be encoded. Both the genetic algorithm and simulated annealing evaluate the encoded solutions and use previous ones to develop new solutions through operations discussed earlier. An example of this encoding for similar mission scenarios can be found in the work of [1, 4, 5]. The representation of possible solutions for this scenario is more complex than found in some applications. Each possible solution must represent the assignment of two types of vehicles to targets with multiple tasks to be performed on them, and thus the encoding scheme chosen has three rows for each solution. The first row contains numbers from the set V = {1, 2, …, N_v} and represents the vehicles that do the non-jamming tasks of classify, attack, and verify. The second row represents the targets and the three non-jamming tasks to be completed on each. This row contains numbers from the set T = {1, 2, …, N_t}. The third row contains numbers from the set J = {N_v +1, …, N_j } which represent the jamming vehicles. Fig. 3 shows an example of a feasible solution for the scenario of three targets each with three tasks being prosecuted by four non-jamming vehicles and two jamming vehicles.

Non-Jamming Vehicles	4	3	1	3	2	1	4	2	1
Targets	1	2	1	3	1	2	2	3	3
Jamming Vehicles	0	0	5	0	0	6	0	5	0

Fig. 3. Example Solution for Electronic Warfare Mission Scenario. This is one possible solution when there are four non-jamming vehicles, two jamming vehicles, and three targets.

Since there are three targets in the example, the second row contains only the numbers 1, 2, and 3. Since each target has exactly three non-jamming tasks to be performed on it, there are three of each number in the row. The first occurrence of a number indicates the classify task on that target, the second indicates the attack task, and the third occurrence represents the verify task. The third row has nonzero entries only under each attack task of the target row. The rest of the entries are zero.

The order in which the tasks are done is taken into account by the objective function. A timeline of the mission is pictured below in Fig. 4. Notation on each bar consists of vehicle number, task, target number, and duration of fly time/task time.

The prosecution plan represented by the feasible solution in Fig. 3 and pictured in Fig. 4 is as follows:

- Vehicle 4 classifies target 1 and then verifies target 2.
- Vehicle 3 classifies target 2 and then classifies target 3.
- Vehicle 5 jams target 1 and then target 3.
- Vehicle 6 jams target 2.
- Vehicle 1 attacks on target 1 and then attacks target 2 and then verifies target 3.
- Vehicle 2 verifies target 1and attacks target 3.

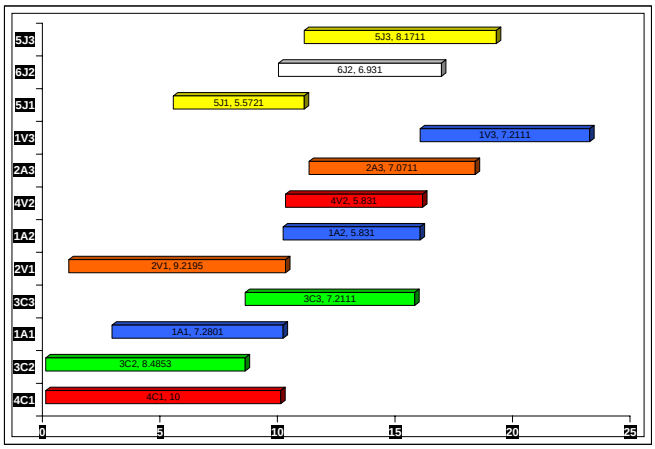

Fig. 4. Timeline for Electronic Warfare Solution. This timeline shows the mission solution that is given in Fig. 3.

E. Counting Feasible Solutions

There are many possible solutions of the type described in the last section. Equation (1) below gives an upper bound for the number of possible solutions with three rows as discussed above. However, other rules need to be applied to determine whether the actual solution is feasible for the EW scenario.

$$N_f = (N_v)^{N_t N_m} * \frac{(N_t N_m)!}{N_{m,1}! N_{m,2}! \ldots N_{m,N_t}!} * (N_j)^{N_a} \qquad (1)$$

Proof:

The number of possible strings for the first row of the solution is the number of ways to fill the $N_t N_m$ (number of targets * number of tasks) places with numbers from the set V = {1, 2, …, N_v}. There are $(N_v)^{N_t N_m}$ ways to do this.

The second row of the solution has the numbers in the set T = {1,2, …, N_t} repeated for as many tasks that are left on the target. This is counted using a permutation with repetitions formula.

$$\frac{(N_t N_m)!}{N_{m,1}! N_{m,2}! \ldots N_{m,N_t}!} \qquad (2)$$

The third row of the solution has zeros and the numbers from the set J = {N_v + 1, …, N_v + N_j }. The nonzero numbers can only be placed under the attack tasks in the second row. If N_a represents the number of attacks in the second row then the number of ways these can be positioned

is $(N_j)^{N_a}$

F. The Objective Function and Operations

For both the genetic algorithm and simulated annealing an objective function must be developed to evaluate the final mission time of a feasible solution. The objective function value is used to determine the fitness of a solution to reproduce in the genetic algorithm and the determination of whether a new solution is accepted or rejected in the simulated annealing algorithm.

Given a feasible solution, the objective function value of that solution represents the total time it takes for the mission to be completed using the task assignment defined by that solution. The objective function must consider task precedence, determine the final time for each task to be completed, and then takes the maximum of all these tasks. Equation 2 shows the objective function to be minimized where t_f is the final overall mission time and $t_j^{(k)}$ is the final time to complete task *k* on target *j*.

$$J = t_f = \max_{j \in T, k \in M} \{t_j^{(k)}\} \qquad (2)$$

For the both the genetic algorithm and simulated annealing there are some common operations that can be employed to create new solutions based on old solutions. These operations include mutate and swap. There are other operations that are used for the genetic algorithm and not for the simulated annealing. They include inversion and crossover. These operations are discussed in the Table 1 below.

TABLE 1
OPERATIONS TO CREATE NEW SOLUTIONS

Operation	Description	Used for GA	Used for SA
Mutation	One element of the row is randomly changed	Yes on vehicle row	Yes on vehicle row
Swap	Two randomly chosen elements in a row are exchanged	Yes on target row	Yes on target row
Inversion	Two random positions are chosen and the elements between these position are inverted	Yes on target row	No
Crossover	Two solutions are combined by choosing a random position and then taking the first part of one solutions row and combining it with the second part of another solutions row	Yes on vehicle row	No

G. New Generations in the Genetic Algorithm

For the genetic algorithm, the process begins by creating a set of feasible solutions called a population (or first generation) and then creates new generations based on the previous one. Each of the newly generated solutions within the generation is evaluated using the objective function to determine the total time it will take for the mission to be completed using the task assignment defined by the solution. Fitness values are then assigned linearly to transform the objective function value to a measure of relative fitness. The selection algorithm selects individual solutions for reproduction based on their relative fitness. Using linear scaling, the expected number of offspring is approximately proportional to the individual's performance. The selection process then uses the roulette wheel method that probabilistically selects individuals from a given generation for reproduction based on their fitness.

The reproductive process, for the genetic algorithm in our problem domain, combines both a crossover operation for the non-jamming vehicle row in one generation and an inversion operation for the target row in another generation. The two other operations applied are elitism and mutation. Elitism ensures that the solution to the problem is monotonically decreasing. The mutation changes one element of the solution at random and is only applied to a small percentage of elements to insure diversity of the population.

The general crossover function takes two feasible solutions and splits the non-jamming vehicle row at some position *n* less than the length of the solution and exchanges the positions 1…*n* in one solution with the 1…*n* entries in the other solution. This action is appropriate for the non-jamming vehicle row of the solution. Fig. 5 and Fig. 6 below show the rows of the solution before and after the crossover operation. The bottom rows of the new solutions remain the same as the original.

Non-Jamming	1	4	2	3	2	1
Targets	2	1	1	2	1	2
Jamming	0	0	5	6	0	0

Non-Jamming	3	1	3	2	4	2
Targets	2	2	1	1	2	1
Jamming	0	6	0	5	0	0

Fig. 5. Original two feasible solutions before the crossover operations is applied.

Non-Jamming	1	4	3	2	4	2
Targets	2	1	1	2	1	2
Jamming	0	0	5	6	0	0

Non-Jamming	3	1	2	3	2	1
Targets	2	2	1	1	2	1
Jamming	0	6	0	5	0	0

Fig. 6. Two new feasible solutions created by the crossover of the original solutions in Fig. 4.

The inversion operation, used on the target row, takes all elements of a single solution between position m and n (m < n) and inverts them. This allows the target row to change but retain the correct number of target entries that are required for a feasible solution. In the example below in Fig. 7 and Fig. 8 it can be seen that the elements in positions two through four are inverted.

Non-Jamming	1	4	2	3	2	1
Targets	2	1	1	2	1	2
Jamming	0	0	5	6	0	0

Fig. 7. The original solution before the inversion operation is applied.

Non-Jamming	1	4	2	3	2	1
Targets	2	2	1	1	1	2
Jamming	0	0	5	6	0	0

Fig. 8. The new solution with positions two through four inverted.

H. Creating Neighbors in the Simulated Annealing

For simulated annealing, the first feasible solution is created at random and evaluated using the objective function. Given any feasible solution *x*, the simulated annealing algorithm creates another solution "close" to *x*, which is based on *x*, to compare to the current best solution. The performance of the algorithm depends critically on this neighborhood operation [6]. If the choice of a neighborhood is too small then the simulated process will not be able to move around quickly enough to find the minimum in a reasonable time. If the neighborhood is too large then the process is essentially a random search.

Goldstein in [6] provides general guidelines for a neighborhood system which are described below.

Let S be a finite set. For each $s \in S$, there is a neighborhood of s, $N_s \subset S$, that satisfies the following:

1. $\forall\, s \in S, s \in N_s$
2. $\forall\, s, t \in S$, $s \in N_t$ if and only if $t \in N_s$
3. $\forall\, s, t \in S$, $|N_s| = |N_t|$
4. $\forall\, s, t \in S$, there exists an integer m and $u_1, u_2, \ldots, u_m$ in S such that $s \in N_{u_1}, u_i \in N_{u_{i+1}}, u_m \in N_t$ for $i = 1,2,\ldots,m\text{-}1$.

The operations used to create a "close" or "neighbor" solution are the mutation or the swap. For the vehicle row of the solution, the mutation is used. This operation simply chooses one of the vehicle numbers in the row at random and changes the number to another possible vehicle number. This creates a different, but feasible solution that is a "neighbor" of the first one. Another operation that is used to produce a "neighbor" solution is the swap operation for the target row. In this operation, two randomly chosen elements in the target row are exchanged. This operation also produces a feasible solution that is based on the previous solution and allows all permutation of that row to be attainable.

I. The Temperature Function for Simulated Annealing

For simulated annealing, a variety of temperature or cooling schedules may be used. After the initial solution is created the algorithm then iterates through a loop, decrementing temperature at each pass. A simple cooling schedule, $T_{i+1} = 0.98T_i$, was first implemented. Although this function was effective it had the potential to allow for conversion of the algorithm to local minima. The second cooling schedule implemented slowed down the convergence and caused the temperature function to, at some times, rise instead of being monotonically decreasing. The cooling schedule is shown in Fig. 9.

```
N=21
if T<=(1-0.2*k)*T
T=(1+0.05*k) T
k=k+1
end

if N<=20
N=N+1
elseif N>20
T=0.98*T
N=21
end
```

Fig. 9. The cooling schedule for the simulated annealing algorithm. This schedule allows for increases in the temperature at some points to avoid converging to a local minima.

The graph in Fig. 10 shows that at some points the temperature actually raises. Since the temperature is used to determine when a worse solution will be accepted (at higher temperatures it is more likely to accept a worse solution) this give more opportunity to get out of local minima.

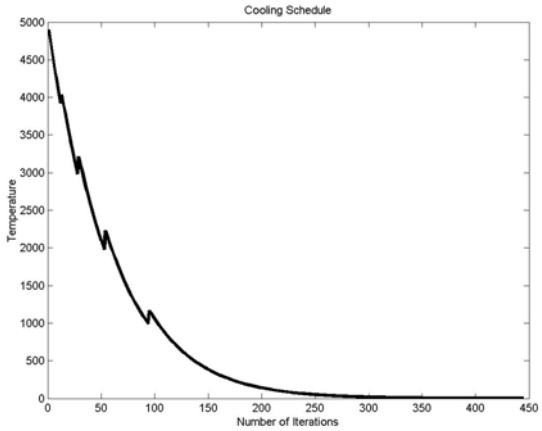

Fig. 10. The graph of the cooling schedule used for the simulated annealing shows the temperature increases at some points. The purpose of this is to avoid converging to a local minima.

III. Performance Analysis

To compare the performance of the algorithms two types of tests were performed: (1) comparison of solutions quality in a given time and (2) comparison of how time to converge to a solution. For both tests, 5000 runs were completed and averaged. For comparison purposes, both the genetic

algorithm and simulated annealing were implemented in MATLAB and a table of estimated flight times was used as input to the algorithm.

The first test compared the total mission time for the best solution found at 0.1, 0.3, 0.5, 0.7, or 0.9 seconds. For the EW scenario, the objective function to be minimized represents the final mission time, thus the smaller the solution, the better the solution. The results are displayed in Fig. 11, Fig. 12, Fig. 13, Fig. 14, and Fig. 15 below for a varying numbers of non-jamming vehicles, targets, and jamming vehicles.

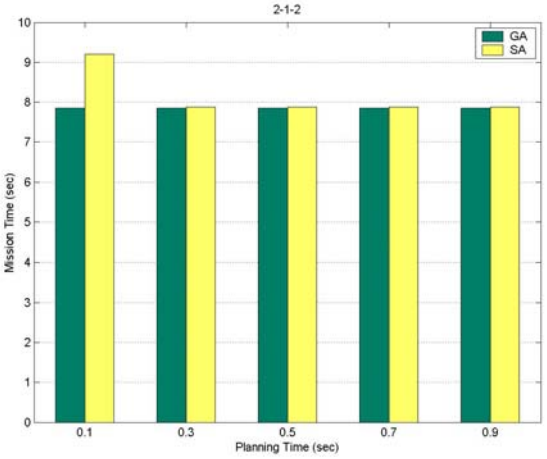

Fig. 11. The graph compares the performance of the genetic algorithm to the simulated annealing on two non-jamming vehicles, one target and two jamming vehicles (2-1-2) mission scenario.

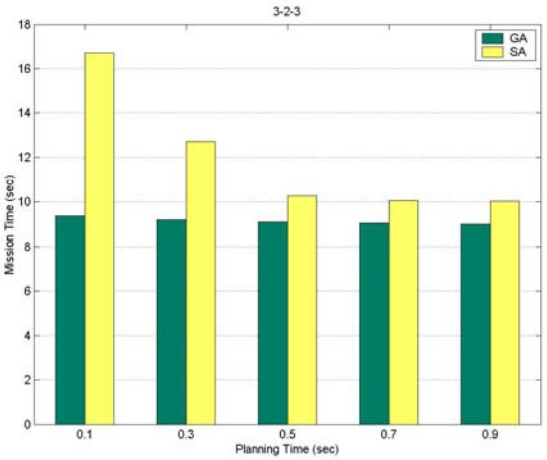

Fig. 12. The graph compares the performance of the genetic algorithm to the simulated annealing on a three non-jamming vehicles, two targets, and three jamming vehicles (3-2-3) mission scenario.

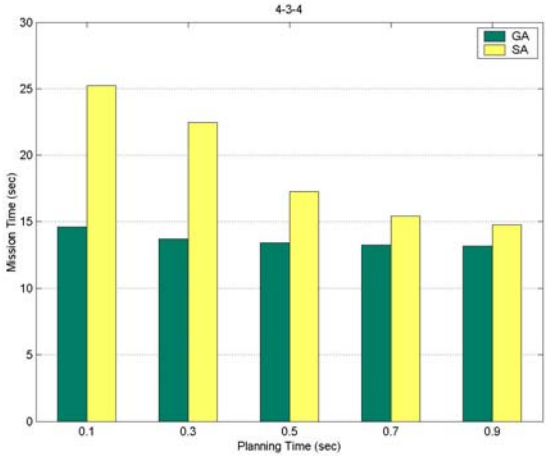

Fig. 13. The graph compares the performance of the genetic algorithm to the simulated annealing on a four non-jamming vehicles, three targets, and four jamming vehicles (4-3-4) mission scenario.

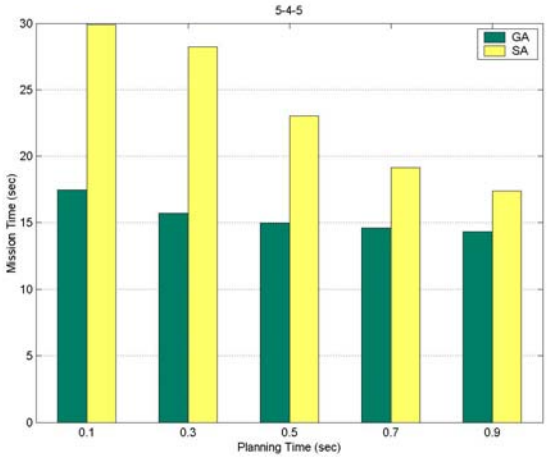

Fig. 14. The graph compares the performance of the genetic algorithm to the simulated annealing on five non-jamming vehicles, four targets, and five jamming vehicles (5-4-5) mission scenario.

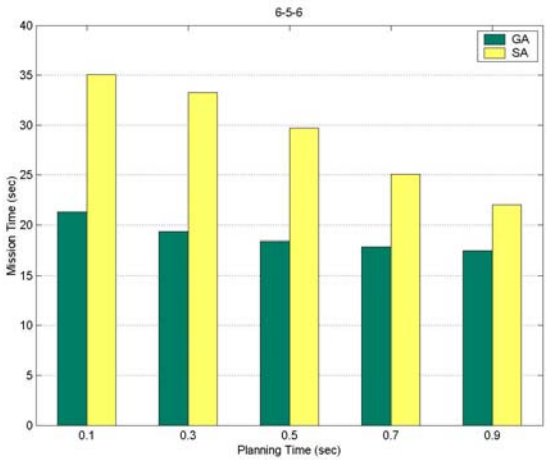

Fig. 15. The graph compares the performance of the genetic algorithm to the simulated annealing on a six non-jamming vehicles, five jamming vehicle, six targets mission scenario.

Fig. 11 indicates that for 2 non-jamming vehicles, 1 targets, and 2 jamming vehicles (2-1-2) scenario the genetic

algorithm and simulated annealing solutions are very close. It is evident in the following figures that once the scenarios get larger the solutions found by the genetic algorithm in a fixed amount of time are superior to the simulated annealing solutions.

The second test compared the time it took each algorithm to converge and the solutions to which they converged. Here, convergence is defined as having no improvement in the solution for a set number of iterations. The iterations without improvement were set to the same number for both algorithms. Fig. 16 below shows that the genetic algorithm converges more quickly than the simulated annealing to a solution. For example, on the largest scenario it took the genetic algorithm half the time (one second) to converge compared to the simulated annealing algorithm (two seconds). In addition to the quicker convergence, the graph in Fig. 17 shows that the objective function value, total mission duration, for the genetic algorithm is always less than the value to which the simulated annealing converges.

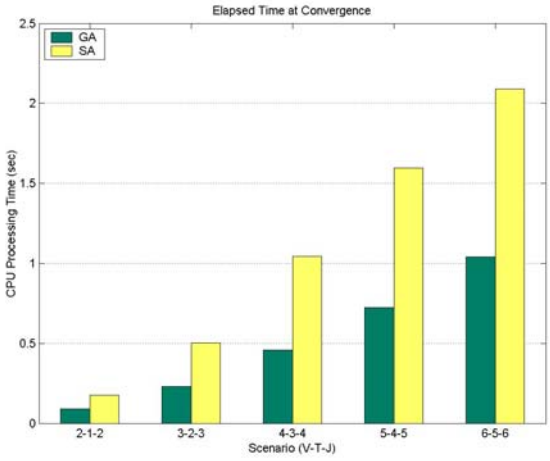

Fig. 16. The graph depicts the time for convergence (no improvement in the solution for the last 200 solutions). The genetic algorithm always converges faster than the simulated annealing algorithm.

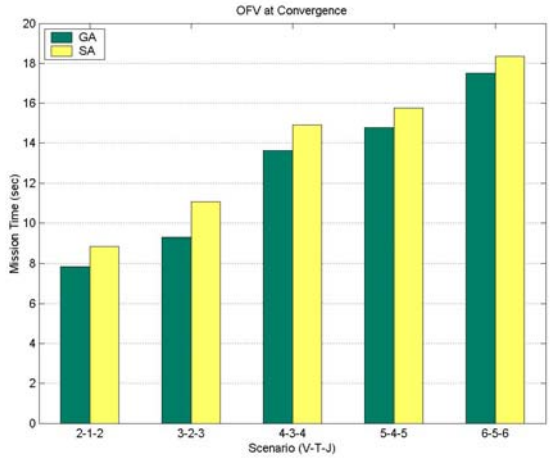

Fig. 17. The graph depicts the objective function value at convergence. The objective function value, total mission duration, is always less for the genetic algorithm.

IV. Conclusion

In this paper a genetic algorithm and a simulated annealing algorithm were both applied the task allocation of multiple UAVs for an electronic warfare mission scenario. After encoding the problem and implementing the algorithms in Matlab, two different types of tests were performed to compare the two algorithms. The performance analysis was based on the quality of solutions in a fixed period of time and time to convergence. For our implementations, all tests indicate that the genetic algorithm is a superior algorithm to the simulated annealing algorithm for this problem domain. The algorithms were implemented using standard practice for each, however tuning of the algorithms may produce somewhat different results.

Acknowledgment

The authors would like to thank Lance Walp who did the original implementation of the genetic algorithm and simulated annealing. Tiffany Brewer and Ben Mitchell are also recognized for modifications made to the genetic algorithm and simulated annealing algorithms for the electronic warfare scenario.

References

[1] T. S. Shima, S. J. Rasmussen, and A. G. Sparks, "UAV Cooperative Control Multiple Task Assignments using Genetic Algorithms," in *Proceedings of American Control Conference*, Portland, OR., 2005.

[2] W. H. Press, S. A. Tenkolsky, W. T. Vetterling, and B. P. Flannery. 1992. (Numerical Recipes in C: The Art of Scientific Computing. Available online at http://library.lanl.gov/numerical/bookcpdf.html

[3] J. Strecker, D. Byrnes, and J. Breitenbucher., "The Simulated Annealing Group Assignment (SAGA) Application," in *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, 2003.

[4] T. S. Shima, and C. Schumacher, "Assignment of Cooperating UAVs to Simultaneous Tasks using Genetic Algorithms," in *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, California, 2005.

[5] M. Darrah, W. Niland, B. Stolarik, and L. Walp, "UAV Cooperative Task Assignments for a SEAD Mission using Genetic Algorithms," in *Proceedings of Guidance, Navigation and Control Conference*, 2006.

[6] L. Goldstein and M. Waterman., "Neighborhood Size in the Simulated Annealing Algorithm," *American Journal of Mathematical and Management Sciences*, vol. 8, pp.409-423, 1988.